We describe an unsupervised, language-independent spelling correction search system. We compare the proposed approach with unsupervised and supervised algorithms. The described approach consistently outperforms other unsupervised efforts and nearly matches the performance of a current state-of-the-art supervised approach.

Introduction

Over two thirds of misspelled queries are caused by transformation errors (insertion, deletion, replacement, and inversion; Li, Duan, & Zhai, 2012; Pollock & Zamora, 1984). Spelling-correction approaches must address these common transformation errors but many cannot without training data. For example, the USHMM1 has a document collection comprising 13 languages. The collection is too large for the low volume of queries to be used for training. Worse, should a supervised approach be deployed, the model might overfit to frequently queried languages, biasing against the results for minority languages.

Although there are many kinds of spelling-correction algorithms, three prominent types are:

- Supervised approaches: Recent supervised approaches (Li et al., 2012) are quite powerful. However, when training data are unavailable or incomplete, a supervised approach cannot be used. For complete evaluation, we compare the effectiveness of this recent work with our own, although the approaches are designed for different environments.
- Unsupervised approaches: Earlier efforts (Aljlayl & Frieder, 2002; Aqeel, Beitzel, Jensen, Grossman, & Frieder, 2006; Gey & Oard, 2001) demonstrated that efficient, simple, unsupervised, rules-based approaches are quite effective. More recently advanced n-gram solutions were demonstrated as only somewhat inferior to supervised approaches (Duan & Hsu, 2011) in particular cases. However, n-grams consider only substrings of sequential characters, and rules-based approaches do not leverage the power of n-grams. We build on the findings of this research, correcting for these limitations by crafting a rule-based approach and using n-grams should our confidence be low.
- Phonetic approaches: Soundex (Mokotoff, 2007) and D-M Soundex are two seminal phonetic algorithms that group like-sounding characters into bins. However, their encoding schemes introduce several problems: dependence on initial letter (Patman & Shaefer, 2003), noise intolerance (Patman & Shaefer, 2003), poor precision (Mitton, 1996; Snae & Bruckner, 2009), and ignoring consonants (Beider & Morse, 2008).

In general, supervised algorithms outperform unsupervised algorithms, particularly in cases in which context is important in correcting a word (Lim, 2012); however, they cannot be used in the absence of training data. We describe an unsupervised approach that has no dependence on domain, language structure, or sequential windows (Soo, 2013; Soo & Frieder, 2010). The proposed solution outperforms prior unsupervised solutions and is comparable with a leading supervised approach.

Approach

Our proposed solution, shown in Figure 1, is an unsupervised algorithm that works as follows:

The user submits a query, $q$, with terms $t_1 \ldots t_n$, thus $q = t_1 \ldots t_n$. If $t_i \in I$, where $I$ is an inverted index, we move...
to the next $t_j \in q$, where $j \neq i$. Otherwise, $t_i \notin I \Rightarrow t_i$ is misspelled or nonexistent. Assuming the former, and a correct spelling of $t_i$ follows:

$$q' \leftarrow \emptyset;$$

for $t_i \in q$ do
  if $t_i \in I$ then
    \[ q'[t_i] \leftarrow t_i; \]
    next;
  end
for $j \in R$ do
  $C_j^i = \emptyset$;
  $k = 0$;
  while $k < K$ do
    \[ C_j^i \leftarrow C_j^i \cup r_k^j(t_i); \]
    $k \leftarrow k + 1$;
  end
end
$C^i \leftarrow \bigcup_{j=1}^{R} \{ C_j^i \};$
$T_n \leftarrow \max_{v \in O^i} \{ v / \sum_{j=1}^{R} |C_j^i| v_j \};$
if $T_n < \beta$ then
  $N^i \leftarrow$ produce n-grams candidates;
  $T_n \leftarrow$ n-grams confidence;
  if $T_n < T_n$ then
    $C^i \leftarrow N^i$
  end
end
$q'[t] \leftarrow \max_{v \in O^i} \{ v|_{C^i} \};$
end
return $q'$;

FIG. 1. Segments algorithm.

At this point, we have selected our most confident candidate sets, $C_t$, for each term $t_i$ from the user's original query $q$. We now select the most confident candidates $t_i \in C_t$, where confidence is measured in the same way as the previously described $T$. The result is our most confident permutation of candidate strings, $q = t_i \ldots t_n$.

**Substring-Generation Rules**

We studied many generation rules and experimentally found the following six to be the most suitable. The primary contributions from these rules are, (a) unlike n-gram solutions, they generate candidates based on nonsequential character windows, which increases the correction rate of our approach, and (b) they have no dependence on language or domain.

For simplicity of illustration, in the following rule definitions, $\alpha(x)$ searches against the index $I$, creates the associated candidate sets $C_t$, and returns the input term $x$. Furthermore, we forego illustrating verification of string length to prevent runtime errors. The percentage symbol (%) represents a wildcard indicating zero or more arbitrary characters. For illustration purposes, the derived substrings for the search term *Mississippi* are given in Table 1.

We evaluated the proposed approach over four data sets, specifically:

- **Web Query Logs**, a subset of the Microsoft Research annotated TREC 2008 Million Query Track data set used by Li et al. (2012). This collection supports comparison with their work and evaluates in context-aware situations. Our selection processes have been outlined further by Soo (2013).
- **USHMM Names**, selected names from the *Yizkor* book collections and victim records (Amir, 2001). It is a collection of 13 different languages, allowing evaluation of the proposed approach on different and diverse languages.

At this point, we have selected our most confident candidate sets, $C_t$, for each term $t_i$ from the user's original query $q$. We now select the most confident candidates $t_i \in C_t$, where confidence is measured in the same way as the previously described $T$. The result is our most confident permutation of candidate strings, $q = t_i \ldots t_n$.

**Substring-Generation Rules**

We studied many generation rules and experimentally found the following six to be the most suitable. The primary contributions from these rules are, (a) unlike n-gram solutions, they generate candidates based on nonsequential character windows, which increases the correction rate of our approach, and (b) they have no dependence on language or domain.

For simplicity of illustration, in the following rule definitions, $\alpha(x)$ searches against the index $I$, creates the associated candidate sets $C_t$, and returns the input term $x$. Furthermore, we forego illustrating verification of string length to prevent runtime errors. The percentage symbol (%) represents a wildcard indicating zero or more arbitrary characters. For illustration purposes, the derived substrings for the search term *Mississippi* are given in Table 1.

$$\begin{align*}
\forall_{i} r_i^1(t_i) &= \left\{ \begin{array}{ll}
\alpha(t_i [2..n-1]) & \text{if } k < K \\
\alpha(t_i [2..n-1]) & \text{if } k = K
\end{array} \right. \\
\forall_{i} r_i^2(t_i) &= \left\{ \begin{array}{ll}
\alpha(t_i [1..(n/2)-1]) & \text{if } k < K \\
\alpha(t_i [1..(n/2)-1]) & \text{if } k = K
\end{array} \right. \\
\forall_{i} r_i^3(t_i) &= \alpha(t_i [1] \% t_i [n]) \\
\forall_{i} r_i^4(t_i) &= \alpha(t_i [1..2]) \% t_i [n-1..n]
\end{align*}$$
USHMM Spoken Names Query Log,\textsuperscript{3} foreign name user query logs. Unlike other data sets, which are synthetically altered (USHMM Names and USCB Names), these contain errors generated by users.

USCB Names, a collection of names, sorted by frequency, according to the results of the U.S. Census Bureau (USCB) during their 1990 collection.\textsuperscript{4} This evaluates the proposed approach on a reasonably large name collection.

Statistical descriptions of each data set can be found in Table 2.

**Query Generation**

Since the USHMM Names and USCB Names data sets do not contain, at least by definition, misspellings, we synthetically generated them. That is, we used four functions that, given a query and a magnitude, return the query with a transformation error of the magnitude 1–4. For example, passing a query *jason* and a magnitude of 2 to the insertion function may return *jXasonL*. All indexes and characters manipulated are randomly selected. No checks are done to determine whether the resulting query is a valid term. However, if a valid term is generated, all evaluated approaches will be equally affected, because all approaches are modified to first search for an exact match. Such instances were not observed.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Search candidates</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>%ississipp%</td>
</tr>
<tr>
<td>2</td>
<td>Missi%ssippi</td>
</tr>
<tr>
<td>3</td>
<td>%ssippi</td>
</tr>
<tr>
<td>4</td>
<td>Mississ%</td>
</tr>
<tr>
<td>5</td>
<td>M%ii</td>
</tr>
<tr>
<td>6</td>
<td>Mi%ii</td>
</tr>
<tr>
<td>k</td>
<td>k = 1</td>
</tr>
<tr>
<td></td>
<td>k = 2</td>
</tr>
<tr>
<td></td>
<td>k = 3</td>
</tr>
</tbody>
</table>

### Methods

Each data set contains multiple tuples comprising a misspelled query and a target query. For all of our experiments, we provide the misspelled query to each approach, and then record whether the target query was returned and in what ranked position. If the misspelled query was synthetically generated from the target query, we run the experiment three times, reporting the average of the runs. We then report our findings as follows.

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Mean</th>
<th>Min</th>
<th>Max</th>
<th>Median</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>USHMM Spoken Names</td>
<td>249</td>
<td>9.80</td>
<td>5</td>
<td>28</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>USHMM Names</td>
<td>656</td>
<td>63.77</td>
<td>7</td>
<td>164</td>
<td>54</td>
<td>60</td>
</tr>
<tr>
<td>USCB</td>
<td>88,799</td>
<td>7.83</td>
<td>3</td>
<td>14</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>Web Query Log</td>
<td>93</td>
<td>29.98</td>
<td>6</td>
<td>60</td>
<td>27</td>
<td>46</td>
</tr>
</tbody>
</table>

*Fig. 2.* $K$ is directly tied to precision and recall.

*Table 1.* Substring-generation rules applied to Mississippi. Each column is an rule iteration.

*Table 2.* Data set statistics.

\textsuperscript{3}http://cs.georgetown.edu/~jason/data/yizkor
\textsuperscript{4}http://www.census.gov/genealogy/www/data/1990surnames/dist.all.last
Web Query Logs results are reported by F1score for comparison with prior work. The remaining results are reported using three metrics:

- Percent found: If, given a misspelling, the correct query was returned in the top 60 results, the query was found. Percentage found is the ratio (number of queries found)/(total queries attempted).
- Rank: The average rank of the target term within the rank-sorted candidate set returned from each query.
- Common rank: The average rank evaluated strictly on the subset in which both n-grams and the proposed approach find the target term. This measure compares the average rank of the proposed approach (often worse overall because of it finding harder terms) with the average rank of 3-grams.

**Results**

The Web Query Log results presented in Table 3 illustrate F1 scores at three different levels: All shows the score for all transcription error queries; CF and CD represent the scores for context-free (84%) and context-dependent (16%) queries, respectively. Context-dependent queries incorporate surrounding words for corrections. For example, a query capital is correct, but capital hill is not (capitol hill would be correct). The former represents CF and the latter CD. Intuitively, the learning approach does outperform the proposed approach. Interestingly, though, it only has an overall F1 score of 0.005 higher. Furthermore, the proposed approach does considerably better with CF corrections than CD ones.

Table 4 displays the results from our USHMM Spoken Names Query Log evaluation. In all measurements, the proposed approach shows statistically significant ($p < 0.01$) performance increases and never has worse results than n-grams.

Table 5 shows results from the USCB and USHMM Names data sets. The data sets are separated by a column with a double line. We measure the percentage found (%) and rank for both 3-grams (3G) and the proposed approach (US). The rank columns have comma-separated values of rank and common rank. Each row represents the average of three tests for the specified query generation function. The result of the better performing approach is italicized. The proposed approach has the best results in all but three instances in which the 3-grams solution supports a slightly better common rank than the proposed approach.

**Conclusions**

Greater than half of all spelling errors today consist of transformation errors. Supervised approaches handle these problems well. However, when training data are insufficient,
supervised approaches fail. We propose an unsupervised search approach in misspelled environments that has no dependence on language structure. We evaluated this against phonetic, supervised, and unsupervised approaches and found favorable results. It has the highest percentage found compared with phonetic and unsupervised approaches and has an F1 score within 0.005 of a modern supervised algorithm. Furthermore, it almost always yields the most favorable rank. We demonstrate the ability of the proposed approach to operate on collections that contain hundreds of thousands of records, with only a scarce few cases in which the rank is negatively affected. The approach described is in use within the archives section of the United States Holocaust Memorial Museum.

References


